PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 9/455 A1

(11) International Publication Number: WO 97/50031

(43) International Publication Date: 31 December 1997 (31.12.97)

(21) International Application Number: PCT/US97/10341

(22) International Filing Date: 17 June 1997 (17.06.97)

(30) Priority Data:
08/669,772 27 June 1996 (27.06.96) US

(71) Applicant: INTEL CORPORATION [US/US]; 2200 Mission College Boulevard, Santa Clara, CA 95052 (US).

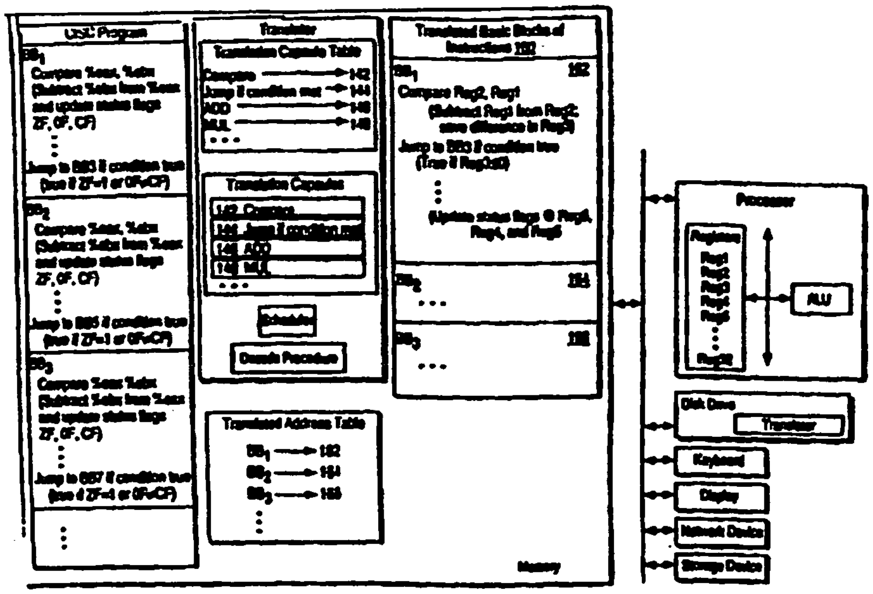
(72) Inventors: FARBER, Yaron; 26 Derech Ha-Yam, 34741 Haifa (IL). LEVHARI, Yossi; 11 Morad HaNachal Street, P.O. Box 238, 40295 Hofit (IL). BARAZ, Leonid; 81 HaOren Street, 28000 Kiryat Ata (IL). LADIRAY, Gallia; 12 Dan Street, 93509 Jerusalem (IL).

(74) Agents: TAYLOR, Edwin, H. et al.; Blakely, Sokoloff, Taylor & Zafman LLP, 7th floor, 12400 Wilshire Boulevard, Los Angeles, CA 90025 (US).

(81) Designated States: AL, AM, AT, AT (Utility model), AU (Petty patent), AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, CZ (Utility model), DE, DE (Utility model), DK, DK (Utility model), EE, EE (Utility model), ES, FI, FI (Utility model), GB, GE, GH, HU, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SK (Utility model), SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: METHOD FOR INCREASING PERFORMANCE OF BINARY TRANSLATED CONDITIONAL INSTRUCTIONS


USC Program
BB1
Compare %eax, %ebx
(Subtract %ebx from %eax and update status flags ZF, OF, CF)
Jump to BB3 if condition true (true if ZF=1 or OF≠CF)
BB2
BB3
Jump to BB7 if condition true
Translator
Translation Capsule Table
Compare
Jump if condition met
ADD
MUL
142
144
146
148
Translation Capsules
Scheduler
Decode Procedure
Translated Address Table
Translated Basic Blocks of Instructions 160
Compare Reg2, Reg1
(Subtract Reg1 from Reg2; save difference in Reg3)
Jump to BB3 if condition true (true if Reg3≤0)
(Update status flags @ Reg4, Reg5, and Reg6)
162
164
166
Memory
Processor
Registers
Reg1
Reg32
ALU
Disk Drive
Translator
Keyboard
Display
Network Device
Storage Device

(57) Abstract

A method for increasing the performance of binary translated conditional instructions (162). According to one embodiment of the invention, a conditional instruction compatible with the first ISA is decoded (BB1). The condition of the conditional instruction is dependent on at least one status flag (ZF=1). The conditional instruction is translated to be compatible with a second ISA (144), wherein the condition of the conditional instructions is altered to be dependent on a previously computed difference between two values (162), the difference residing in a memory location (REG3).

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav	TM	Turkmenistan
BF	Burkina Faso	GR	Greece		Republic of Macedonia	TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's	NZ	New Zealand		
CM	Cameroon		Republic of Korea	PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

METHOD FOR INCREASING PERFORMANCE OF BINARY TRANSLATED CONDITIONAL INSTRUCTIONS

FIELD OF THE INVENTION

The invention relates to computer systems, and, in particular, to methods for increasing the performance of certain types of instructions.

BACKGROUND OF THE INVENTION

Most often, computer programs are initially written in high-level program statements. In order to be executed by a computer, the program statements are compiled into machine instructions that a microprocessor can recognize and execute. The machine instructions are selected from a set of machine instructions unique to a particular Instruction Set Architecture (ISA).

In the field of ISAs, there have arisen two fundamentally different types of instruction sets: the Complex Instruction Set Computer architecture (CISC) and the Reduced Instruction Set Computer architecture (RISC). The CISC architecture utilizes an instruction set that allows for complicated and flexible ways of calculating such elements as memory addresses. One of the defining characteristics of the CISC architecture is the use of variable length instructions, which adds additional complexity to the decoding hardware. Although very powerfully, all the complexity of the CISC instruction set usually requires additional clock cycles to execute each machine instruction.

A RISC processor, on the other hand, is designed to operate more efficiently by processing a relatively smaller set of instructions. The RISC design is based on the premise that most of the instructions a computer decodes and executes are simple. As a result, RISC architecture limits the number of instructions that are built into the microprocessor, and optimizes each so it can be carried out very rapidly, usually within a single clock cycle.

It is possible for computer program statements that have been compiled into machine instructions for a CISC processor to be translated in order to be executed on a

RISC processor. This translation process is normally carried out by a binary translator, which typically consist of a set of instructions stored in memory.

The RISC machine executes the translated instructions by using the RISC instruction set to emulate how the untranslated instructions would be executed on a CISC processor. Practitioners in the art will understand, that when a RISC processor emulates a CISC program, many of the inherent performance advantages of the RISC processor are missed.

Consider, for example, the conditional instruction. A conditional instruction performs an operation, such as transferring the flow of execution from one instruction to another, when a specified condition is true. If the condition is false, the conditional instruction is treated as a no-operation (i.e., an instruction having no effect on the machine's current architectural state).

A conditional instruction is typically based on the results of a previously executed arithmetic instruction, such as the compare instruction. A compare instruction, when executed on a CISC processor, compares two values by subtracting one value from a second value and updates a set of status flags based on the difference of the two values. The actual difference generated by the comparison, however, is not saved in a memory location (i.e., is architecturally transparent).

The status flags are typically represented by one-bit fields in a dedicated register. In one prior art CISC ISA, the status flags include a Zero Flag (ZF) indicating the difference is 0, a Sign Flag (SF) representing the most significant bit of the difference, and an Overflow Flag (OF) indicating the difference is too large or too small to be represented in the number of bits allocated to store the result.

The conditional instruction, when executed on a CISC processor, evaluates a logical combination of the status flags in order to determine whether the condition of the conditional instruction is met. In one example, the conditional instruction evaluates the complex expression ($ZF = 1$ or $SF != OF$).

The RISC microprocessor, however, typically has no native support for setting status flags (i.e., dedicated registers). Therefore, a RISC processor emulates a CISC processor executing a compare instruction by allocating bit fields within a general purpose register for the setting of status flags.

To emulate a conditional instruction evaluating the set of status flags, the RISC processor employs a series of instructions provided in the RISC instruction set. Use of the series of RISC instructions, however, misses the performance advantage of the RISC processors. Typically, the instruction set of a RISC processor includes one conditional instruction that efficiently evaluates a single value generated by a previously executed compare instruction.

Therefore, it can be appreciated that there is a need for a method of translating conditional instructions that are originally dependent on a logical combination of status flags, to be dependent on an actual difference generated by a previously executed compare instruction.

SUMMARY AND OBJECTS OF THE INVENTION

The present invention provides a method for increasing the performance of binary translated conditional instructions. According to one embodiment of the invention, a conditional instruction compatible with a first ISA is decoded. The condition of the conditional instruction is dependent on at least one status flag.

The conditional instruction is then translated to be compatible with a second ISA, wherein the condition of the conditional instruction is altered to be dependent on a previously computed difference between two values.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example and not limitation in the figures of the accompanying drawings, in which like references indicate similar elements, and in which:

Figure 1 illustrates a computer system capable of implementing one embodiment of the present invention.

Figure 2 is a flow diagram describing the steps in one embodiment of the present invention.

DETAILED DESCRIPTION

A method is described for increasing the performance of binary translated conditional instructions executed on a computer system that has no native support for setting status flags.

The present invention translates instructions from a source architecture instruction set that includes support for status flags, to a target architecture instruction set that has no native support for status flags. For example, the source architecture instruction set may be the instruction set of a CISC, Very Long Instruction Word computer (VLIW), or a RISC architecture having native support for status flags, and the target architecture instruction set may be the instruction set of a RISC, or other computer architecture that has no support for status flags.

Each conditional instruction translated by the present invention is typically included in a separate basic block of instructions (BB). The basic block of instructions are translated one at a time.

A basic block of instructions typically includes a contiguous sequence of non-branch machine-instructions ending with a conditional branch instruction. Most computer programs, such as the CISC program of Figure 1, consists of multiple basic blocks of instructions stored in a physical static sequence (e.g., BB_1, BB_2, BB_3 . . .)._The basic blocks of the CISC program shown in Figure 1 are instructions compiled to be executed on a CISC program.

The instructions of each basic block of instructions in the CISC program are translated sequentially by the translator shown in the memory of Figure 1. In alternative embodiments, the translator of the present invention may be stored on other computer-readable mediums, including magnetic and optical disk, and other random-access memory devices, accessible via the disk drive shown in Figure 1.

During the translation process each instruction is first decoded into an opcode and operands by the decode procedure shown stored in the translator of Figure 1. A decoded instruction is then translated by one of the translation capsules shown in the translator.

A hash value of a decoded instructions opcode is used to index the Translation Capsule Table shown in the translator. The table in turn provides the address of a translation capsule corresponding to the opcode of the decoded instruction. For example, the opcode for the compare instruction may hash to an entry containing the address value 142 for the compare instruction's translation capsule, as shown in Figure 1.

The translation capsule that corresponds to a decoded instruction's opcode provides a sequence of instructions from the target architecture instruction set in place of the decoded instruction's opcode. In addition, the capsule function translates the memory addresses of the decoded instruction's operands to be compatible with the memory addresses of the target architecture.

CISC program instructions translated by the translator are stored in the area of memory allocated for storing Translated Basic Blocks of instructions 160 shown in Figure 1. These instructions are compatible with the target architecture instruction set.

Figure 2 is a flow diagram describing the steps performed to increase the performance of translated conditional instructions according to one embodiment of the present invention. In block 202, a program counter identifies a basic block of untranslated instructions.

In block 204, a first instruction from the basic block of instructions is decoded by the translator into an opcode and operands. The opcode indicates an operation to be performed by the instruction, and the operands identify a register, cache, or memory location storing data to be operated on by the opcode.

In one embodiment of the present invention, the instruction is decoded by a procedure that includes a separate sequence of instructions stored in the translator. In alternative embodiments. the instructions can be decoded by a procedure stored in memory outside the translator or provided in a hardware component.

In block 206, the translator selects a translation capsule function that corresponds to the opcode of the decoded instruction by hashing the opcode to get an index value from the Translation Capsule Table as described above. The capsule function is selected from a set of translation capsule functions stored in_the translator. Each capsule function provides a separate sequence of instructions in place of the decoded instructions opcode in order to emulate the execution the decoded instruction on the target computer architecture. The instructions provided from the capsule function are selected from the instruction set of the target architecture.

In addition, the capsule functions translate the memory addresses of the decoded instruction's operands to be compatible with the memory addresses of the target architecture. The translation may consist of adding the decoded instructions operands to an offset base address of the target architecture.

In one embodiment, the translation capsules are indexed in a table by a hash value of the opcodes from the source computer architecture instruction set (e.g., the 12 least significant bits of each opcode). The hash value of the decoded instructions opcode is then used to determine the location of the corresponding translation capsule in the table. Each entry in the table includes the address of the translation capsule corresponding to the opcode, and may include other information regarding the translation capsule, such as whether the opcode takes memory operands.

In block 208, the selected translation capsule function is called and receives the decoded instructions operands and opcode as its parameters. In response, the capsule provides a series of instructions in place of the decoded instructions opcode, translates the memory addresses of the decoded instructions operands, and stores the results in an area of memory allocated to store translated basic blocks of instructions.

If the instruction being translated is a compare instruction, in block 210 the respective capsule function provides a series of instructions that generate a difference between two values and save the difference in a memory location. The memory location may include a general register or an address space of a cache on a processor, for when the translation is performed dynamically (i.e., during execution of a program), or the memory location may be an address space on a disk, for when the translation is performed statically (i.e., when program is not being executed).

For example, a compare instruction compiled to be executed on the source architecture may include the operations of (subtract register %ebx from register %eax and update status flags ZF, OF, and CF), as shown in BB1 of the CISC program in Figure 1. On the other hand, a compare instruction translated by the present invention may include the operations of (subtract reg1 from reg2 and save difference in reg3), as shown in translated basic block of instructions 162 in Figure 1.

In one embodiment, an additional set of instructions is provided by the respective capsule function to emulate setting the status flags as part of the translated compare instruction. The flag updating operations are included because subsequent instructions in the program may depend on the settings of the flags. In one embodiment, the flags are emulated by allocating bit fields within a general purpose register.

In alternative embodiments, other arithmetic instructions that update status flags based on the results of a data operation may be translated in a similar manner.

If the instruction being translated is a conditional instruction, in block 212 the respective capsule function provides a series of instructions in order to have the condition of the translated conditional operation depend directly on the difference generated by the most recently executed translated compare instruction. The instructions provided by the function reference the memory location where the difference generated by the most recently executed compare instruction has been saved. No status flag evaluating instructions, however, are provided.

For example, the condition of a conditional instruction compiled to be executed on the source architecture may include the operations of (true if ZF =1 or, OF != CF), as shown in BB1 of the CISC program in Figure 1. On the other hand, the condition of a conditional instruction translated by the present invention may include the operation (true if reg3 ≤ 0), as shown in translated basic block of instructions 162 in Figure 1.

Typically, computer architectures that have no native support for status flags provide a conditional instruction, in their instruction set, that depends on a single value stored in a register. Therefore, conditional instructions translated by the present invention benefit from better utilization of the more efficient target computer architecture.

In decision block 214, it is determined whether additional machine instructions remain in the basic block of instructions to be translated. If any instructions remain to be translated, the remaining instructions are translated in steps similar to those described in blocks 204-212.

Once the basic block of instructions has been translated, an address for the translated basic block of instructions can be entered into a translated address table. As a result, when the basic block of instructions is called to be executed, the translated address table will provide the address of the translated basic block of instructions.

In block 216 the translated instructions of the basic block of instructions are re-scheduled by the translator to be executed by a processor in an order based on inter-instruction dependencies, which is commonly referred to as the critical path of execution. In one embodiment, the critical path of execution includes the instructions of a translated compare instruction which generates and saves a difference between two values in a memory location. These instructions are followed by the translated conditional instruction which evaluates the difference in the memory location.

The instructions of the translated compare instruction that update the status flags are scheduled to be executed by a processor after the critical path of execution, as shown in the translated basic block of instructions 162 in Figure 1. As a result, the critical path of executing the translated compare and the translated conditional instruction is shorter and faster than it would be if it included the instructions to update the status flags.

In an alternative embodiment, the instructions of the translated compare instruction may be scheduled to save the difference to a memory location and update the status flags in parallel when the present invention is performed on a computer system having a processor that is able to execute multiple instructions per clock cycle. This type of processor is commonly referred to as a superscalar processor.

The parallel execution is performed by identifying a group of instructions that do not conflict with each other, or require use of the same functional unit. These instructions are then scheduled to be executed in the same clock cycle.

In the foregoing specification, the invention has been described with reference to specific exemplary embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention. For instance, the present invention may be performed dynamically during the execution of a computer program or statically when the program is not being executed. Moreover, where it has been shown that the binary translator, CISC program, translated basic blocks, and translated address table are stored in memory, these items and instructions can also be stored on other computer-readable mediums, including magnetic and optical disk, and other random-access memory devices. The specification and drawings are, accordingly, to be regarded in an illustrative rather than restrictive sense.

CLAIMS

What is claimed is:

1. A computer-implemented method for translating a conditioned instruction compatible with a first Instruction Set Architecture (ISA) into a conditional instruction compatible with a second ISA, the method comprising the steps of:

a) decoding a conditional instruction compatible with the first ISA, wherein a condition of the conditional instruction is dependent on at least one status flag; and

b) translating the conditional instruction to be compatible with the second ISA, wherein the condition of the conditional instruction is altered to be dependent on a previously computed difference between two values.

2. The method of claim 1, wherein the previously computed difference between two values is stored in a memory location.

3. The method of claim 2, further including the following steps prior to the step of decoding a conditional instruction:

c) decoding an arithmetic instruction compatible with the first instruction set, the arithmetic instruction when executed, generates the difference between two values and updates said status flag, wherein the difference is architecturally transparent to the first ISA; and

d) translating the arithmetic instruction to be compatible with the second ISA, wherein the arithmetic instruction is altered to have the arithmetic instruction when executed generate the difference between two values and place the difference in the memory location.

4. The method of claim 3, wherein the arithmetic instruction is a compare instruction.

5. The method of claim 4, wherein the method is performed during an execution of a computer program that includes the arithmetic instruction and the conditional instruction.

6. The method of claim 5, wherein the first ISA is a Complex Instruction Set Computer Architecture, and the second ISA is a Reduced Instruction Set Computer Architecture.

7. The method of claim 6, further including the step of:

e) scheduling the arithmetic instruction to have the arithmetic instruction update the status flag outside a critical path of execution.

8. The method of claim 6, further including the step of:

e) scheduling the arithmetic instruction to have the arithmetic instruction update the status flag and save the difference in the memory location, in parallel.

9. . A computer-implemented method for translating conditional instructions to increase a performance of conditional instructions when executed on a computer architecture exclusive of native support for status flags, the method comprising the steps of:

decoding an arithmetic instruction, the arithmetic instruction, which when executed, generates a difference between a first value and a second value and updates at least one status flag, wherein the difference is transparent to the computer architecture exclusive of native support for status flags;

translating the arithmetic instruction to have the arithmetic instruction when executed generate the difference between the first value and the second value, wherein the difference is placed in a memory location;

decoding a conditional instruction, wherein a condition of the conditional instruction is dependent on the status flag; and

translating the conditional instruction to have the condition of the conditional instruction dependent on the difference placed in the memory location.

10. A computer-implemented method for translating a first set instructions compatible with a first Instruction Set Architecture (ISA) into a second set of instructions compatible with a second ISA, the method comprising the steps of:

a) decoding an arithmetic instruction compatible with the first ISA, the arithmetic instruction which, when executed, generates a difference between two values and updates a status flag, wherein the difference is architecturally transparent to the first ISA; and

b) translating the arithmetic instruction to be compatible with the second ISA, wherein the arithmetic instruction is altered to have the arithmetic instruction when executed generate the difference between two values and place the difference in a memory location.

11. A binary translator comprising:

means for decoding a conditional instruction compatible with a first ISA, wherein a condition of the conditional instruction is dependent on at least one status flag; and

means for translating the conditional instruction to be compatible with a second ISA, wherein the condition of the conditional instruction is altered to be dependent on a previously computed difference between two values.

12. The binary translator of claim 11, wherein the previously computed difference is residing in a memory location.

13. The binary translator of claim 12, further comprising:

means for decoding an arithmetic instruction compatible with the first ISA, the arithmetic instruction which when executed, generates the difference between two values and updates said status flag, wherein the difference is architecturally transparent to the first ISA; and

means for translating the arithmetic instruction to be compatible with the second ISA, wherein the arithmetic instruction is altered to have the arithmetic instruction when executed generate the difference between two values and place the difference in a memory location.

14. The binary translator of claim 13, wherein the arithmetic instruction is a compare instruction.

15. The binary translator of claim 14, wherein the means for translating the arithmetic instruction and the means for translating the conditional instruction, translate the respective instructions during an execution of a computer program that includes the arithmetic instruction and the conditional instruction.

16. The binary translator of claim 15, wherein the first ISA is a Complex Instruction Set Computer Architecture, and the second ISA is a Reduced Instruction Set Computer Architecture.

17. The binary translator of claim 16, further comprising:
a means for scheduling the arithmetic instruction to have the arithmetic instruction update said status flag outside a critical path of execution.

18. The binary translator of claim 16, further comprising:
a means for scheduling the arithmetic instruction to have the arithmetic instruction update the status flag and save the difference at the memory location in parallel.

19. A computer readable-medium having stored thereon a plurality of a sequence of instructions for translating instructions, the sequence of instructions, which when executed by a processor, cause the processor to perform the steps of:
a) decoding a conditional instruction compatible with a first instruction set architecture (ISA), wherein a condition of the conditional instruction is dependent on at least one status flag; and
b) translating the conditional instruction to be compatible with a second ISA, wherein the condition of the conditional instruction is altered to be dependent on a previously computed difference between two values.

20. The computer readable-medium of claim 19, wherein the difference between the two values is residing in a memory location

21. The computer readable-medium of claim 20, wherein the plurality of sequences of instructions further include additional instructions, which when executed by the processor, cause said processor to perform the steps of:
c) decoding an arithmetic instruction compatible with the first instruction set, the arithmetic instruction which when executed, generates the difference between two values and updates said status flag, wherein the difference is architecturally transparent to the first ISA; and
d) translating the arithmetic instruction to be compatible with the second ISA, wherein the arithmetic instruction is altered to have the arithmetic instruction when

executed generate the difference between two values and place the difference in the memory location.

22. The computer readable-medium of claim 21, wherein the arithmetic instruction is a compare instruction.

23. The computer readable-medium of claim 22, wherein the step of translating the arithmetic instruction and the step of translating the conditional instruction are performed during an execution of a computer program that includes the arithmetic instruction and the conditional instruction.

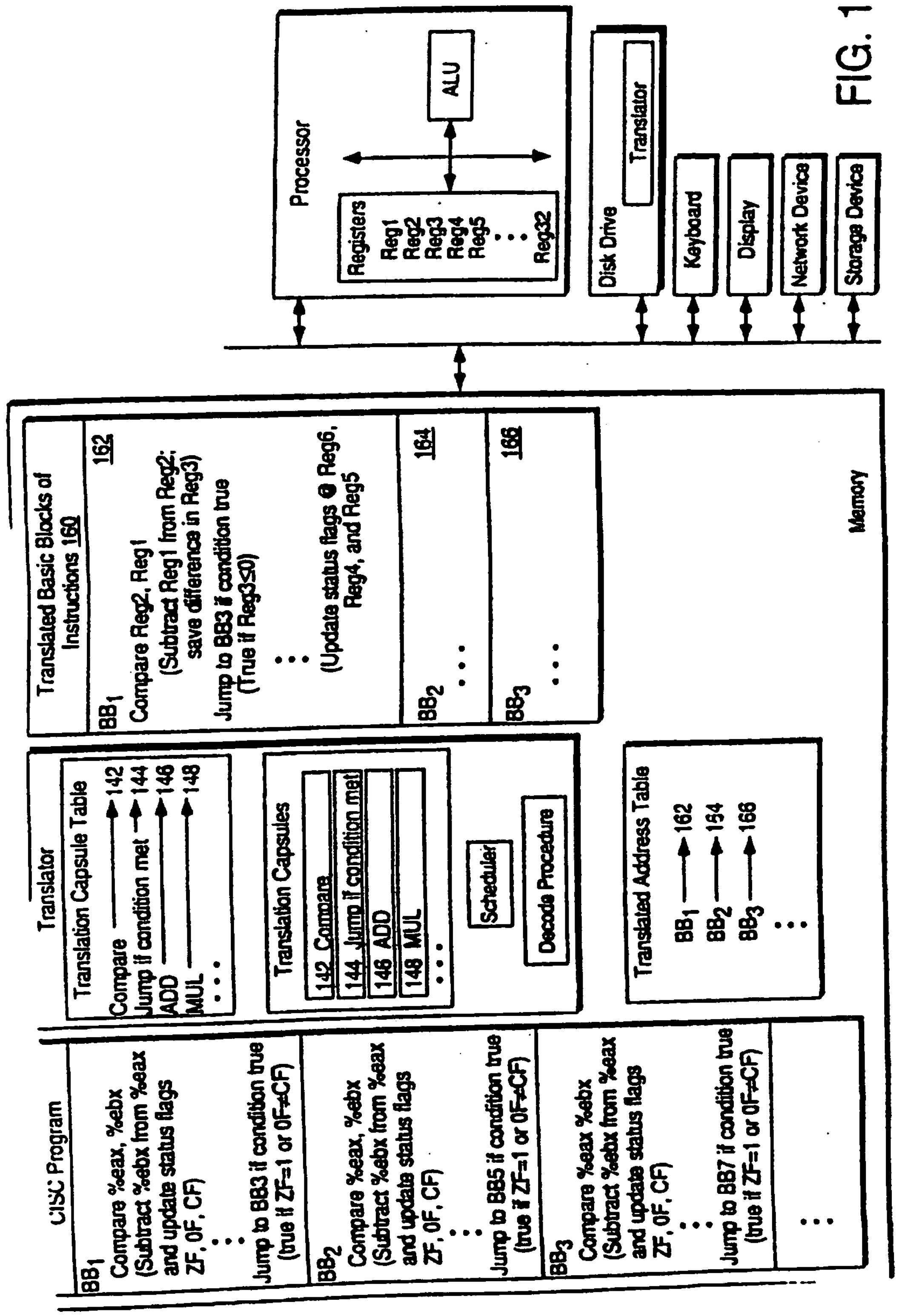
24. The computer readable-medium of claim 23, wherein the first ISA is a Complex Instruction Set Computer Architecture, and the second ISA is a Reduced Instruction Set Computer Architecture.

25. The computer readable-medium of claim 24, wherein the plurality of sequences of instructions further include additional instructions, which when executed by the processor, cause said processor to perform the step of:

e) scheduling the arithmetic instruction to have the arithmetic instruction update said status flag outside a critical path of execution.

26. The computer readable-medium of claim 24, wherein the plurality of sequences of instructions further include additional instructions, which when executed by the processor, cause said processor to perform the step of:

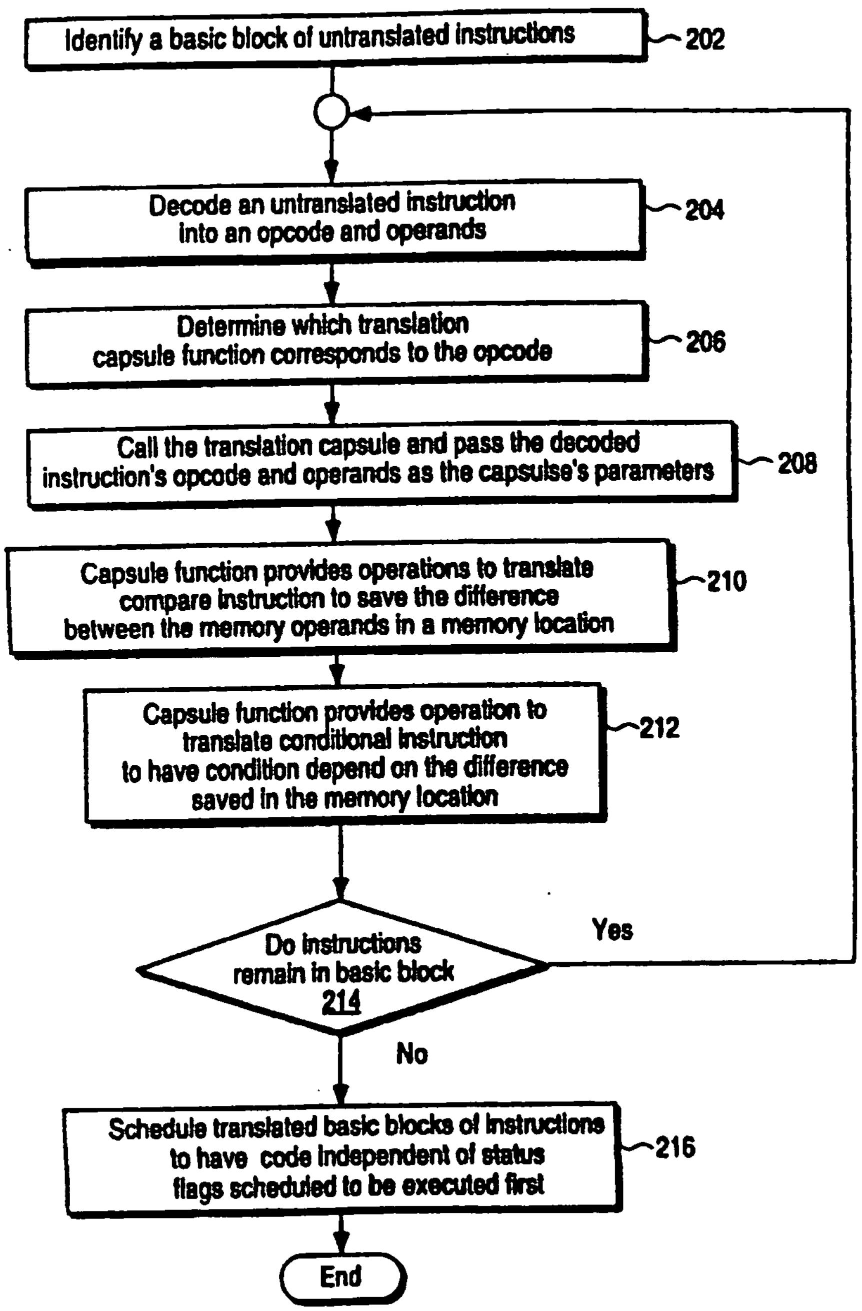
e) scheduling the arithmetic instruction to have the arithmetic instruction update the status flag and save the difference in a memory location, in parallel.



Processor
ALU
Registers
Reg1
Reg2
Reg3
Reg4
Reg5
• • •
Reg32
Disk Drive
Translator
Keyboard
Display
Network Device
Storage Device
Memory
CISC Program
BB1
Compare %eax, %ebx
(Subtract %ebx from %eax and update status flags ZF, 0F, CF)
• • •
Jump to BB3 if condition true (true if ZF=1 or 0F≠CF)
BB2
Compare %eax, %ebx
(Subtract %ebx from %eax and update status flags ZF, 0F, CF)
• • •
Jump to BB5 if condition true (true if ZF=1 or 0F≠CF)
BB3
Compare %eax %ebx
(Subtract %ebx from %eax and update status flags ZF, 0F, CF)
• • •
Jump to BB7 if condition true (true if ZF=1 or 0F≠CF)
• • •
Translator
Translation Capsule Table
Compare → 142
Jump if condition met → 144
ADD → 146
MUL → 148
• • •
Translation Capsules
142 Compare
144 Jump if condition met
146 ADD
148 MUL
• • •
Scheduler
Decode Procedure
Translated Address Table
BB1 → 162
BB2 → 164
BB3 → 166
• • •
Translated Basic Blocks of Instructions 160
BB1
162
Compare Reg2, Reg1
(Subtract Reg1 from Reg2; save difference in Reg3)
Jump to BB3 if condition true (True if Reg3≤0)
• • •
(Update status flags @ Reg6, Reg4, and Reg5
BB2
164
• • •
BB3
166
• • •

FIG. 1

FIG. 2


Identify a basic block of untranslated instructions
202
Decode an untranslated instruction into an opcode and operands
204
Determine which translation capsule function corresponds to the opcode
206
Call the translation capsule and pass the decoded instruction's opcode and operands as the capsulse's parameters
208
Capsule function provides operations to translate compare instruction to save the difference between the memory operands in a memory location
210
Capsule function provides operation to translate conditional instruction to have condition depend on the difference saved in the memory location
212
Do instructions remain in basic block 214
Yes
No
Schedule translated basic blocks of instructions to have code independent of status flags scheduled to be executed first
216
End

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US97/10341

A. CLASSIFICATION OF SUBJECT MATTER
IPC(6) :G06F 9/455
US CL :395/705
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 395/385, 586, 705, 707, 708, 709

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

APS (Fogg, translate, object, code, program, architecture, instruction, conditional)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,313,614 A (GOETTELMANN et al.,) 17 MAY 1994, see abstract, fig. 11, col. 3 line 55 to col. 4 line 50.	1-26
A	US 5,301,325 A (BENSON) 05 APRIL 1994, see abstract.	1-26
Y	US 5,241,678 A (FUTAMURA ET AL.) 31 AUGUST 1993, see abstract, col. 2 line 1 to col. 3 line 16.	1-26
A	US 4,951,195 A (FOGG, JR. ET AL.) 21 AUGUST 1990, see abstract.	1-26
A	MOTOROLA. MC68030 Enhanced 32-bit Microprocessor User's Manual, Second Edition, 1989, pg. 3-71 to 3-72.	1-26

[✓ X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date		
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed	"&"	document member of the same patent family

Date of the actual completion of the international search 08 OCTOBER 1997	Date of mailing of the international search report 12 NOV 1997
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer RICHARD ELLIS [illegible] Telephone No. (703) 305-9600

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US97/10341

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	GAINES, R. Stockton. On the Translation of Machine Language Programs, Communications of the ACM, Volume 8, Number 12, December 1965, pp. 736-741.	1-26
A	BERGH, Arndt B. et al. HP3000 Emulation on HP Precision Architecture Computers, Hewlett-Packard Journal, December 1987, pp. 87-89.	1-26
A	HUNTER, Colin et al. DOS at RISC, Byte Magazine, November 1989, pp. 361-368.	1-26
A	SAARI, Michael. 68000 Binary Code Translator, FORML Conference Proceedings, 1987, pp. 48-52.	1-26
A	BANNING, John. The XDOS Binary Code Conversion System, IEEE, 1989, pp. 282-287.	1-26